RELAZIONE PROGETTO GESTIONE DI RETI

Claudio Burrafato mat.520333

Email: c.burrafato@studenti.unipi.it

Il progetto consiste nel aggiungere delle statistiche stampate dal tool di analisi ndpiReader contenuto in nDPI, dove vengono raccolti i nomi al dominio dei flussi:

* flow->ssh_tls.server_info

* flow->host_server_name

e vengono inseriti in una UT_hash_table, ordinandoli per numero di occorrenze.

Per far questo, quindi, ho innanzitutto creato una struttura dati che ho chiamato hash_stats; dove vengono usati tre campi: un char* per il nome del dominio usato come chiave della tabella, un int per l'occorenza usato come valore della tabella e l'handle per la UT_hash_table(il campo hh chiamato così per utilizzare le macro della libreria uthash).

Dopodiché, nella funzione help, ho aggiunto alla riga 475 tra le opzioni suggerite per l'argomento v, il 4. Questo perché ho scelto di usare un opzione a parte per vedere, in maniera diretta, cosa veniva stampato dal mio codice.

Fatto questo, alla riga 2313, ho aggiunto il codice della funzione(hash_stats_sort) da utilizzare nella macro della UT_hash,HASH_SORT, per ordinare la tabella in ordine decrescente in base al numero di occorrenze.

In fine ho modificato la funzione printFlowsStats() aggiungendo il caso in cui verbose fosse==4. In questo blocco viene costruita la hash table in cui inserire le statistiche da stampare.

La struct hostsHashT è il puntatore principale della hash table, host_iter viene usato per i vari cicli della macro HASH_ITER, dove vengono memorizzate le struct a mano mano visitate dal ciclo; inoltre nei vari HASH_ITER viene usato il campo della struct hh come viene suggerito dalla guida sulle UT_HASH. Infine ad ogni ITER vengono usati delle struct temporane, chiamate in questo caso tmp e seguite dai vari numeri.

Per riempire la table, viene prima controllato se il dominio è già una chiave presente, e nel caso contrario viene inserita usando la macro HASH_ADD_KEYPTR (viene usata questa in quanto il campo è un char*). Se invece la chiave è già presente, viene incrementato il rispettivo valore che rappresenta il numero di occorenze nel flusso.

Nel caso in cui viene inserita una chiave, si controlla se la lunghezza della table non superi un certo valore(la variabile len_table_max); nel caso in cui questo valore venga superato si cancellano un tot di elementi(la variabile toDelete) per liberare spazio nella table(harvesting).

Dopo gli inserimenti, con HASH_SORT viene ordinata la table utilizzando la funzione definita prima.

Nelle righe successive viene stampato il contenuto della tabella. Nelle righe 2774-2776 ho scritto questo codice in modo da stampare la colonna delle occorrenze in maniera allineata, in quanto mi sono accorto che usando \t della printf, alcuni numeri non venivano stampati allineati.

Nelle ultime righe viene, infine, cancellata la tabella e vengono deallocate le struct che la compongono.

Nel terminale, per compilare ed eseguire ndpiReader, ho usato questa istruzione:

./nDPI/example/ndpiReader -i ../tests/pcap/XXXXX.pcap -v 4. I file .pcap usati per i test sono quelli della cartella tests di nDPI